斜率优化？截距优化！

immortalCO，一只猫

概述

斜率优化是 NOI 难度的一种非常常见的 DP 优化手段，适用于一些转移是线性式子的 1D1D 动态规划。
然而，在一般的式子推导中，使用了一种比较复杂的斜率式，推导难度较大，且并不直观。
我在这里提出了一种新的斜率优化的推导方式，能以一种更加直观的方式推出斜率优化的式子，从而更加直观、高效的分析问题。 本文仅介绍方法，不提供例题，请结合暑假作业一起服用。

式子推导

如果一个 DP，设状态为 `f[i]`，如果能推导出
`f[i] = c[i] + min{a[j] + k[i] * b[j] | j < i 且 满足其他和 i 有关的条件}` （如果是 `max`，可以把符号全部取反变成 `min`），那么这个 1D1D 动态规划可以使用斜率优化进行优化。
然而这个式子看起来并不是特别直观，使用经典方法证明，又比较麻烦。考虑重写式子，给他赋予**几何意义**。首先，`c[i]` 和 `j` 无关，因此我们求的就是 `min(a[j] + k[i] * b[j])`。设 `x[j] = -b[j], y[j] = a[j]`，我们得到了我们要最小化的式子：

`y[j] - k[i] * b[j]`

注意到由斜截式 `y = k * x + b` 可得到 `b = y - k * x`，因此上面这个式子的意义就是**截距**！也就是说，我们的任务是，将所有可以作为转移的 `j` 表示成二维平面上一个点 `p[j] = (x[j], y[j])`，我们需要找一个 `j` 使得经过 `p[j]` 的斜率为 `k[i]` 的直线的截距最小！因此，斜率优化优化的实际上是**截距**!
那么从截距的意义上来看，我们需要找一个点，使得给定斜率 `k` 的直线经过它的截距最小，那么显然我们可以想象有一条斜率为 `k` 的尺子从 y 轴负无穷处向上移动，碰到第一个点时停下来，那么这时候这个尺子的截距就是最小的截距。那么这个点会在哪里呢？显然它会在下凸壳上。而且设这个点到凸壳上上一个点的斜率为 `k1`，到凸壳上下一个点的斜率为 `k2`，我们有 `k in [k1, k2]`。由于下凸壳上斜率递增，也就是说，得到凸壳后，我们只需要在凸壳上二分斜率即可。

处理方式

现在的任务是维护凸壳，分为几种不同的情况。

1. `x[i]` 和 `k[i]` 均单调，没有附加限制
   单调队列维护凸壳，无需二分，用 `l` 作为决策点，时间复杂度线性

2. `x[i]` 单调，但 `k[i]` 不保证单调，没有附加限制
   单调栈维护凸壳，用二分寻找决策点

3. `x[i]` 不单调，有附加限制
   用资磁单点插入的平衡树维护动态凸包，或 CDQ 分治维护凸壳，然后二分或单调性的扫

如果有附加限制，我们可以用线段树套上述算法的数据结构，或进行分治。